Learning Based Wind Farm Control

AI2P Final Report

Eren Aydoslu (4997778)

I. Introduction

A. Problem Statement

This project aims to advance new wind farm (WF) control strategies by leveraging artificial intelligence, improving on existing methods. A key area of improvement lies in wake steering control, a technique that adjusts the yaw angles of turbines to redirect the wakes and optimize power production. Although this strategy is not new, our approach introduces a novel, data-driven method for modeling the wind farm itself, which could enhance the effectiveness of wake steering. Currently, high-fidelity simulators such as SOWFA [4] are widely used to model wind flow, capturing complex interactions, turbine wake effects, and thermodynamic dynamics. However, the computational cost of these simulators is prohibitively high for real-time applications, making them unsuitable for continuous control. In contrast, simplified models like FLORIS [7] offer a more feasible option for real-time control but often sacrifice accuracy, leading to suboptimal performance in terms of power output.

In response to these limitations, this project focuses on two key objectives: (1) developing a deep learning-based model to predict WF behavior at a significantly reduced computational cost relative to high-fidelity simulators, and (2) creating an end-to-end, learning-based controller that uses turbine-level data to adjust configurations dynamically and maximize power output. We propose using Physics-Informed Neural Networks (PINNs) to model wind flow across the wind farm based on the Navier-Stokes equations. By embedding physics laws directly into the learning process, our approach aims to reduce the computational cost relative to high-fidelity simulators while retaining accurate representations of wake dynamics and turbine interactions. To achieve this, we will leverage large-eddy simulation (LES) data from 3D simulations, specifically using 2D horizontal slices at the turbine level, providing sufficient detail at a reduced dimensional complexity.

We aim to develop an end-to-end learning-based controller capable of optimizing turbine configurations in real time by dynamically adjusting yaw angles based on turbine input data. Reinforcement learning (RL) will be employed to train this controller, enabling it to effectively learn strategies for maximizing power output by mitigating wake effects through optimized yaw control.

The LES simulation is performed over a 5000-meter by 5000-meter area and is done over 12000 seconds. There are three different cases with different topologies and each case contains one simulation with greedy control and one with wake steering. The data consists of 300 by 300 grids, where each point in the grid contains information on u, v, and w, the flow velocities in x, y, and z directions respectively. Each grid is separated by 5 seconds, giving us 2400 different grids per simulation. We solely focus on the first case as this should give us enough training points.

B. Motivation

Wind energy plays a crucial role in meeting global sustainability targets, especially within Europe, where wind farms are expected to form a substantial component of the renewable energy mix. However, realizing the full potential of wind farms remains challenging due to the complex dynamics involved in wind flow and turbine interactions. Environmental variables such as atmospheric conditions, combined with the aerodynamic interactions among turbines, create wake effects that can reduce power output, particularly in larger wind farms. Existing control strategies struggle to adapt to these complexities effectively and often require trade-offs between computational feasibility and predictive accuracy. Therefore, there is a critical need for innovative, data-driven methods that can both model wind flow and optimize WF control.

II. Literature Review

Machine learning, particularly deep learning, has advanced rapidly in recent years, with significant successes in the wind industry. Applications include wind power forecasting, wind speed prediction, and wind farm wake modeling [9], [15], [19]. However, the integration of physical laws into the training of deep learning models seems to be largely overlooked in wind energy studies despite this approach gaining traction in other fields [20].

Implicit neural representations refer to neural network models that encode information about signals, shapes, or physical phenomena within the network's parameters and architecture, rather than relying on explicit data representations [3]. These networks learn continuous and differentiable mappings from input coordinates (e.g., spatial or temporal variables) to desired outputs, essentially learning a representation of the data as a continuous function.

Physics-Informed Neural Networks (PINNs) are a specialized form of implicit neural representations that incorporate physical laws directly into the learning process [12]. Unlike traditional neural networks that primarily rely on data for training, PINNs integrate partial differential equations (PDEs) or other governing physical equations into their loss functions.

The integrated loss functions guide the network to "understand" the underlying physics and ensure that representations satisfy physical principles.

Mainly, there are three major applications of PINNs: forward simulation, model inversion, and equation discovery [10]. Forward simulation concerns carrying out the numerical solution of a known physical system given the initial conditions and boundary conditions. Equation discovery involves identifying the underlying mathematical models or differential equations that govern a given set of data in cases where the governing equations are not well understood. However, our primary interest lies in model inversion. Model inversion utilizes PINNs to infer unknown parameters, inputs, or initial conditions of a system from observed data. In other words, the goal in our case is to find the flow velocity values that satisfy the underlying PDE.

In our inverse simulation problem, we employ the non-dimensionalized incompressible Navier-Stokes equations to improve numerical stability. Non-dimensionalization can reduce the number of parameters by scaling the variables and therefore, ease the the problem at hand [14]. Although air is a compressible fluid by nature, the flow velocities in our problem are relatively low, with free stream velocities around 8 m/s. At these speeds, air can be considered incompressible and variations in density should be negligible [8]. This approach allows us to accurately capture the essential flow characteristics while reducing computational complexity, thereby facilitating more effective training and model inversion in our application.

A critical challenge in the application of PINNs is scaling them to larger problem domains. As the problem domain expands or the solution complexity increases, the neural network must grow in size — both in terms of the number of layers and the number of neurons per layer — to maintain an accurate representation of the solution. The increase in network size leads to a higher number of parameters making the optimization harder and the training process more computationally expensive. Additionally, larger domains necessitate a greater number of training points to adequately sample the solution space, further increasing the difficulty [10].

A significant contributor to these scalability issues is the spectral bias inherent in traditional neural networks. Neural networks typically prioritize learning low-frequency components of the solution, resulting in slower convergence and reduced accuracy for high-frequency features [1], [2], [11], [18]. The bias becomes increasingly challenging in large domains, where the low-frequency preference of traditional PINNs struggles to adapt to the finer, high-frequency variations required to represent complex dynamics accurately [10].

To address these challenges we plan on adapting the SIREN structure [13]. SIREN utilizes periodic activation functions, which are shown to be better suited for representing high-frequency details in the solution. Unlike traditional activation functions, sine functions facilitate the learning of complex, oscillatory patterns without requiring an increase in the number of parameters. The i-th layer in SIREN can be expressed as:

$$\phi_i(x_i) = \sin\left(\omega \cdot \mathbf{x}_i^T \theta_i + \mathbf{b}_i\right) \tag{1}$$

where ω i an hyperparameter concerning the frequency of the activation function and is usually set to $\omega = 30$, $\mathbf{x}_i$ is the input vector, θ_i is the layer weights, and $\mathbf{b}_i$ is the bias term.

Specifically, we use H-SIREN proposed by [6] which has been demonstrated to have better performance in fluid flow problems. H-SIREN changes the activation function in the first layer from $\sin(x)$ to $\sin(\sinh(2x))$ in order to better capture higher frequency components. By utilizing H-SIREN, we expect improved accuracy and faster convergence, in our use case of tackling the inversion problem for solving the wind flow over the wind farm.

Finally, we aim to employ the trained PINN as the simulation environment within an RL framework to optimize wind farm control. Specifically, we intend to use the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm proposed by [5] as our RL agent. TD3 is an RL technique designed to handle continuous action spaces, and in our case should generate the optimal yaw actions for each turbine. By using the PINN as an environment model, the RL agent can iteratively test different yaw configurations, simulating their effects on wind flow and wake dynamics over the wind farm. This approach enables us to generate optimized control strategies without the computational expense of running high-fidelity simulations during training.

III. Methods

A. Wind Flow Prediction

The PINN we use is designed to predict the wind flow dynamics across the wind farm based on spatial temporal, and control-related inputs. The inputs to the PINN include the spatial coordinates x, y, and z, along with the time variable t. Additionally, each turbine's yaw angle and the overall wind direction serve as control inputs, allowing the model to capture the effects of turbine configurations and environmental conditions on the flow field. Given these inputs, the PINN outputs the wind flow components u, v, and w, representing the velocity components along the x, y, and z axes, respectively. It also predicts the pressure p at each specified spatiotemporal point. The backbone of the architecture is fully connected layers and we experiment with both using SiLU activation functions and the H-SIREN architecture. The network is trained using a data loss and a physics loss. The data loss helps the network to learn the specific case problem we're solving, whereas the physics loss makes sure the network behaves according to physical laws.

To non-dimensionalize Navier-Stokes equations, we start by introducing characteristic scales for each variable. Let L represent the characteristic length scale, U the characteristic velocity, and $T = L/U$ the characteristic time scale. The pressure can be non-dimensionalized by dividing by ρU^2, where ρ is the density of air, however, pressure is not that relevant to our experiments as we don't have data for it. The non-dimensional variables are defined as:

$$x^* = \frac{x}{L},\ y^* = \frac{y}{L},\ z^* = \frac{z}{L},\ t^* = \frac{t}{L/U}$$
$$u^* = \frac{u}{U},\ v^* = \frac{v}{U},\ w^* = \frac{w}{U}$$

Substituting these non-dimensional variables into the incompressible Navier-Stokes equations give us the following equations. The continuity equation

$$\nabla \cdot \mathbf{u} = 0 \tag{2}$$

becomes

$$\nabla^* \cdot \mathbf{u}^* = 0 \tag{3}$$

and the momentum equation

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = -\nabla p + \mu \nabla^2 \mathbf{u} \tag{4}$$

becomes

$$\frac{\partial \mathbf{u}^*}{\partial t^*} + \mathbf{u}^* \cdot \nabla^* \mathbf{u}^* = -\nabla^* p^* + \frac{1}{\text{Re}} \nabla^{*2} \mathbf{u}^* \tag{5}$$

where, $\nabla^* = L\nabla$, $\mathbf{u}$ is the flow velocity vector, μ is the dynamic viscosity of air, and $\text{Re} = \frac{\rho U L}{\mu}$ is the Reynolds number.

We choose $L = 5000$ meters, representing the wind farm's spatial extent, and $U = 8$ m/s, the free-stream wind velocity, as our characteristic scales. Using $\rho = 1.204$ and $\mu = 1.789 \cdot 10^{-5}$ we get a Reynolds number of $\text{Re} \approx 2.69 \cdot 10^9$ This high Reynolds number indicates that inertial forces dominate over viscous forces in the flow, placing the system well within the turbulent flow regime. Given this extremely high Reynolds number, we assume the Euler regime, where viscous effects are negligible. Consequently, we can assume the fluid in question is inviscid and can use the Euler equations for inviscid flow instead of the full Navier-Stokes equations [16]. Therefore, the momentum equation simplifies to the following:

$$\frac{\partial \mathbf{u}^*}{\partial t^*} + \mathbf{u}^* \cdot \nabla^* \mathbf{u}^* + \nabla^* p^* = 0 \tag{6}$$

To use these equations in the model, we have to define them as loss functions. Consider the momentum equation (equation 6), in a real scenario, fluid behaves according to the equation and therefore satisfies the equation. However, the model's predictions will have errors and will not capture a real scenario perfectly, in other words, in the model's predictions the left-hand side of the equation 6 will not be equal to zero. We can use these deviations as a loss function to guide the model to make the left-hand side of the equation zero. The following are the errors of the model concerning the continuity equation, x-momentum, y-momentum, and z-momentum using the non-dimensionalized variables.

$$e_{\text{con}} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \tag{7}$$

$$e_x = \frac{\partial u}{\partial t} + u \cdot \frac{\partial u}{\partial x} + v \cdot \frac{\partial u}{\partial y} + w \frac{\partial u}{\partial z} + \frac{\partial p}{\partial x} \tag{8}$$

$$e_y = \frac{\partial v}{\partial t} + u \cdot \frac{\partial v}{\partial x} + v \cdot \frac{\partial v}{\partial y} + w \frac{\partial v}{\partial z} + \frac{\partial p}{\partial y} \tag{9}$$

$$e_z = \frac{\partial w}{\partial t} + u \cdot \frac{\partial w}{\partial x} + v \cdot \frac{\partial w}{\partial y} + w \frac{\partial w}{\partial z} + \frac{\partial p}{\partial z} \tag{10}$$

We use automatic differentiation to compute the derivatives of the network outputs (wind velocity components $u,\ v,\ w$, and pressure p) with respect to the inputs $x,\ y,\ z$, and t. This allows us to evaluate the gradients, which are needed for the equations.

Although our data is a 2D horizontal slice at a constant z level, and the z input is constant through training, we can still compute gradients with respect to z due to automatic differentiation. This approach enables us to estimate z-gradients with reasonable accuracy. For instance, consider the continuity equation (equation 7), we can estimate $\partial u/\partial x$ and $\partial v/\partial y$ from the data. By enforcing the continuity condition to be zero, we can then infer the z-gradient $\partial w/\partial z$, allowing us to incorporate vertical flow characteristics even from 2D data. However, it suffices to say that we don't expect the model to learn the flow characteristics at different z levels but to only capture the dynamics at the given vertical level.

Given the errors in continuity and x, y, z-momentums, $e_{\text{cont}}, e_x, e_y, e_z$ respectively, the physics loss function is defined as follows:

$$\mathcal{L}_{\text{con}} = \frac{1}{N} \sum e_{\text{con}}^2 \tag{11}$$

$$\mathcal{L}_x = \frac{1}{N} \sum e_x^2 \tag{12}$$

$$\mathcal{L}_y = \frac{1}{N} \sum e_y^2 \tag{13}$$

$$\mathcal{L}_z = \frac{1}{N} \sum e_z^2 \tag{14}$$

$$\mathcal{L}_{\text{physics}} = \mathcal{L}_{\text{con}} + \mathcal{L}_x + \mathcal{L}_y + \mathcal{L}_z \tag{15}$$

We calculate the data loss ($\mathcal{L}_{\text{data}}$) only on u, v, and w using mean squared error as the simulation data does not contain information on pressure. While the data loss function is calculated using the simulation data, the physics loss is only calculated through the means of automatic differentiation of outputs with respect to the inputs. The total loss function of the network is constructed as follows:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{data}} + \lambda \mathcal{L}_{\text{physics}} \tag{16}$$

where λ is a hyperparameter to tune.

Due to non-dimensionalization the x and y are scaled between 0 and 1. However, time requires additional handling due to its broader range. Given a total simulation time of 12,000 seconds, we first scale t by L/U, which normalizes it to $[0, 19.2]$. To avoid potential instabilities from such a large range, we further scale t to $[-1, 1]$, following the approach suggested in [17]. Specifically, we divide the time input by 9.6 and subtract 1, yielding a stable normalized range for t. Since this transformation affects the temporal gradients, each $\partial/\partial t$ term must also be divided by 9.6 to ensure that the network's temporal derivatives align with the original time scale. This scaling preserves gradient consistency while improving training stability. Additionally, to normalize angular inputs like the wind direction and turbine yaw angles, while preserving their cyclic properties, we convert these angles to their sine and cosine values.

Given that we have 2400 grids of 300x300, that gives us 216 million training samples. For the purposes of wind flow prediction over the wind farm, we train the models only on wake steering simulation otherwise, the training becomes very computationally expensive as we also have some hyperparameters to tune. We can then select the best model to train on both cases to use in the RL environment.

B. Wake Steering Control via Reinforcement Learning

To optimize wake steering in our wind farm control strategy, we use the trained PINN as the environment for an RL setup, where the TD3 agent learns to adjust turbine yaw angles. Instead of predicting absolute yaw angles, the TD3 agent's actions are designed to indicate the deviation of each turbine's yaw angle from the wind direction. For instance, if the agent predicts an action of 0, the turbine faces directly into the wind, maximizing its individual power output but possibly creating suboptimal wake effects for downstream turbines.

We define a maximum yaw angle, which restricts the range of actions the agent can take. This constraint ensures that the yaw adjustment remains within a realistic operational range, where the turbine can deviate up to 30 degrees on either side of the incoming wind direction. For example, if the wind direction is 90 degrees and the defined maximum yaw angle is 30 degrees, the TD3 agent can only set yaw angles between 60 and 120 degrees, corresponding to -30 and +30 degrees of yaw deviation, respectively.

The primary objective of the RL agent is to maximize the mean power production across all turbines in the wind farm. Consequently, we define the reward function as the mean power output of all turbines at each time step, incentivizing the agent to find configurations that not only enhance the power production of individual turbines but also account for the collective power output.

Given that the PINN model includes a time input parameter, we consider two approaches for managing this parameter during RL training. The first approach is to fix the time input to zero throughout each episode. This option is motivated by a desire to minimize potential input combinations that the PINN has not encountered during training, thereby reducing the likelihood of extrapolation errors in the flow predictions.

Alternatively, we can vary the time input by setting it to the current step in the episode, incrementing it with each time step. This approach introduces a temporal aspect into the control strategy, potentially allowing the RL agent to leverage any time-dependent effects that may arise from accumulated changes in yaw configurations. However, this option might be a bit difficult, as time-varying inputs may introduce combinations of spatiotemporal conditions that are far from the training data distribution, potentially affecting prediction accuracy.

IV. Evaluation

A. Wind Flow Prediction

First, let's analyze the impact of different physics coefficients (λ) on the training dynamics of the PINN, which uses the SiLU activation function with a hidden layer size of 128. Figure 2 shows the learning curves for total loss across various values of λ (1, 0.1, and 0.01), while Figure 3 provides insights into the behavior of the physics loss component throughout training.

From Figure 2, we observe that lower physics coefficients ($\lambda = 0.1$ and $\lambda = 0.01$) result in faster convergence and lower total loss values compared to the model trained with $\lambda = 1$. This behavior suggests that reducing the physics coefficient can make the training process easier, leading to more efficient learning of the target solution. Intuitively, one might assume that smaller physics coefficients would result in reduced emphasis on the physics-based regularization term, effectively lowering the challenge of satisfying the physical constraints during training. This relaxation would allow the model to focus more on fitting the data component of the loss, thereby achieving quicker convergence.

However, Figure 3 reveals an interesting picture by showing that the magnitude of the physics loss remains relatively similar across all values of λ. Despite the different physics coefficients, the physics loss does not significantly deviate in scale, indicating that the PINN model still adheres to the physical constraints, regardless of the reduced emphasis on the physics component in the total loss.

In our experiments with the H-SIREN model, training proved to be significantly more challenging than anticipated. Unlike the SiLU-based model, which showed stable convergence across a range of hyperparameters, the H-SIREN model exhibited extreme sensitivity to key parameters such as the frequency parameter (ω), learning rate, and the physics coefficient (λ). These sensitivities made the training process highly unstable, with losses often reaching unreasonably high values or plateauing at suboptimal levels that were notably worse than those achieved by the SiLU model (see Figure 4).

A particular point of difficulty was the selection of ω, which controls the frequency of the sinusoidal activation functions. In the literature, $\omega = 30$ is commonly used for H-SIREN models and typically yields good results in other applications. However, in our case, this value consistently led to poor performance across multiple training attempts, suggesting that the optimal ω for this application differs significantly from conventional values. This discrepancy may be due to the complexity of our physics-informed setup or the specific characteristics of the wind flow problem.

Figure 1 illustrates the SiLU-based PINN model's performance in predicting wake dynamics over the wind farm. The left panel displays the ground truth data from the simulation. The center panel shows the model's predicted output, which seems to replicate the general structure and flow direction of the wakes successfully. The right panel presents the squared error between the simulation data and the model output.

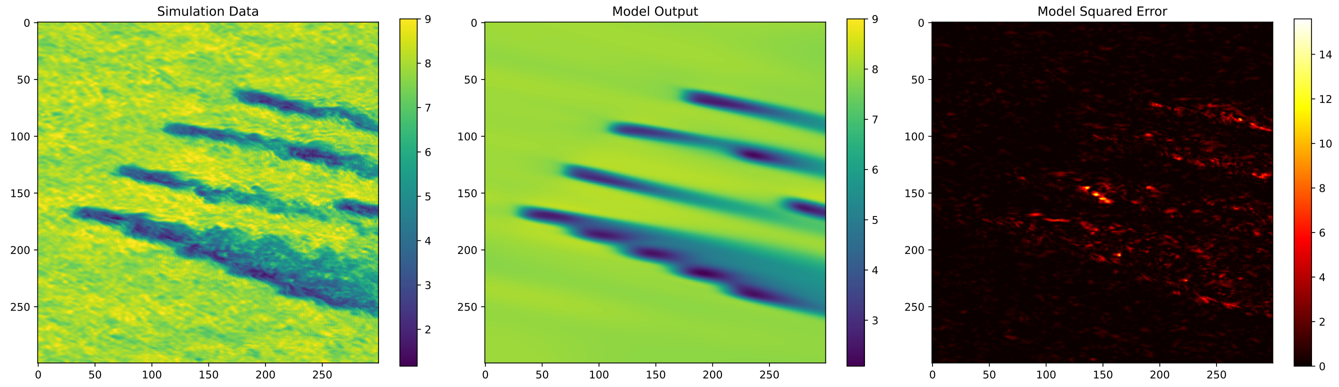

Fig. 1: SiLU model prediction on the test set, showing a comparison between the simulation data (left), the model output (center), and the squared error (right). The visualizations highlight the wake patterns produced by turbines and the model's ability to approximate these features

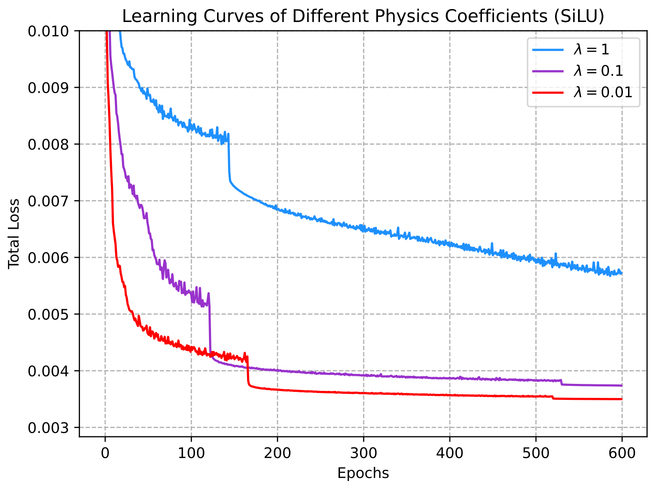

Fig. 2: The effect of physics coefficient λ on the learning curves of the models with SiLU activation function with hidden size 128.

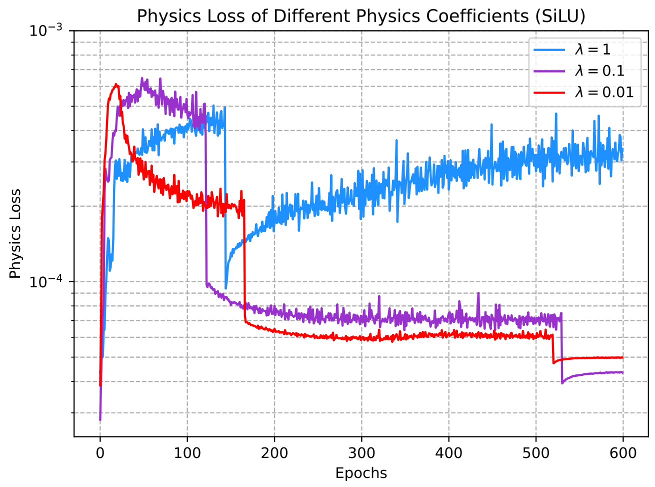

Fig. 3: The physics loss over the course of training in the models with SiLU activation function with a hidden size of 128. The resulting physics losses are after scaling by the physics coefficient.

Model Type	ω	λ	Hidden Size	Test Loss*
SiLU	-	0.01	128	0.2286
SiLU	-	0.01	256	**0.2234**
H-SIREN	10	0.01	256	0.2262

Table 1: The best 3 models according to loss on the test set. *The loss is unnormalized by multiplying by 64. This is due to the fact that wind speeds has been scaled down by a factor of 8. Unnormalization yields a more interpretable value.

Although the SiLU model successfully approximates the major wake features, the error map reveals that small, high-frequency details are challenging for the model to capture fully.

Initially, we anticipated that H-SIREN would outperform the SiLU-based model, as its architecture is designed to better capture high-frequency components and fine-grained details in the solution space. Given the complex nature of wake dynamics, H-SIREN's use of sinusoidal activation functions with a hyperbolic component theoretically positions it well to handle the higher-frequency variations present in the flow. However, despite our expectations, H-SIREN did not outperform the SiLU model, with both models showing very similar performance on the test set.

A likely reason for this outcome is the extreme sensitivity of H-SIREN to hyperparameter choices. Training H-SIREN required precise tuning of the frequency parameter (ω), learning rate, and physics coefficient, making it challenging to identify a stable combination of hyperparameters that could fully leverage H-SIREN's capabilities. It is possible that the performance discrepancy could be reduced or even reversed with further hyperparameter exploration, but within the scope of our experiments, H-SIREN could not consistently achieve a better result.

One advantage observed with H-SIREN, however, was faster training times compared to SiLU. The model reached

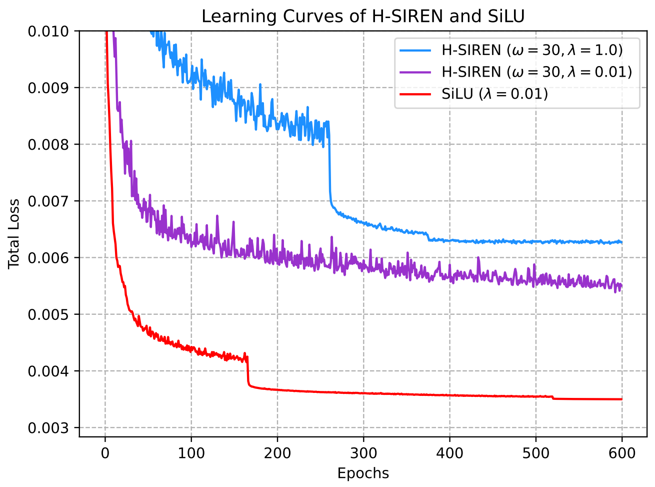

Fig. 4: Learning curves of different H-SIREN configurations compared to SiLU

its convergent behavior more quickly, suggesting that with the right setup, H-SIREN could offer both speed and accuracy benefits. Nonetheless, the challenges in stabilizing H-SIREN training highlight the need for careful hyperparameter optimization when applying this architecture in physics-informed settings.

One limitation of the current models is their dependency on a fixed wind farm topology, which restricts their ability to generalize across different layouts or configurations of turbines. Future work could address this by integrating Graph Neural Networks (GNNs) with the PINN architecture. GNNs are well-suited for representing the relational structure between turbines, capturing spatial dependencies in a way that generalizes across different wind farm topologies. By combining the strengths of PINNs for physics-informed modeling with the flexibility of GNNs for handling varying topologies, we could develop models capable of adapting to diverse wind farm layouts. This approach would enhance the model's applicability in real-world scenarios where wind farm layouts differ significantly, making it possible to deploy learned control strategies across multiple sites with minimal retraining.

B. *Wake Steering via Reinforcement Learning*

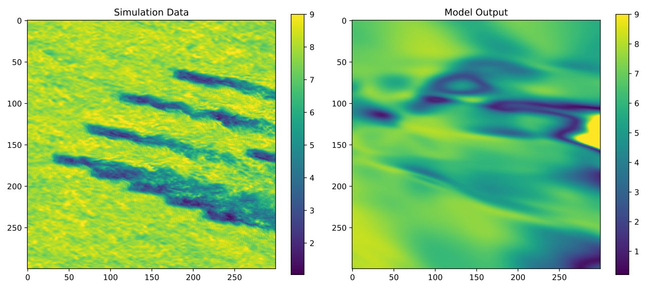

Fig. 5: SiLU prediction with 20 degrees of yaw offset

When using the SiLU-based PINN model as the environment in our RL framework, the results were largely disappointing. One of the primary challenges encountered was that the RL agent's actions would quickly drift outside the distribution of yaw angles the PINN had seen during training. As a result, the PINN model would frequently predict unrealistically high wind speeds in response to these out-of-distribution inputs. See Figure 5 for an example output with unconventional yaw angles.

This behavior created an unintended feedback loop where the RL agent learned to exploit the environment's weaknesses. Specifically, the agent discovered that by assigning unusual yaw angles—often angles that would be impractical or sub-optimal in real-world scenarios—it could achieve artificially high power outputs as a result of the exaggerated wind speeds predicted by the PINN. Consequently, the RL agent's learned policy focused on unconventional angles that maximized power in the flawed simulation, rather than adopting realistic, physics-consistent strategies for improving wake steering.

Although we attempted to mitigate this issue by constraining the maximum yaw angle, the improvement was minimal. While limiting the yaw angle range did reduce some of the extreme predictions, the RL agent still found ways to exploit the environment by selecting unconventional angles that kept producing inflated wind speeds. After a few episodes, the agent's behavior stabilized around repeatedly choosing extreme yaw angles, such as 15, -15, 15, -15, ..., demonstrating a pattern that did not align with optimal wake steering or realistic control strategies.

One promising direction for future work is to explore transfer learning by adapting the PINN model as the RL agent itself. This approach would involve repurposing the trained PINN model by removing its last layer and replacing it with a new output layer designed to generate control actions, such as turbine yaw angles. This modified PINN-RL agent would retain the underlying physics-informed knowledge from its initial training, providing a strong foundation for learning effective control strategies.

The RL environment, in this case, would be a simpler, computationally efficient wind farm model, like FLORIS, which can approximate wake interactions at a lower fidelity than high-fidelity simulators but without the instability issues we observed when using the PINN as an environment. By leveraging the physics-aware insights encoded in the PINN through transfer learning, this approach could yield an RL agent capable of making more realistic control decisions within a stable and computationally feasible environment, advancing our goal of optimizing wind farm performance.

V. Ethics Statement

Our research required extensive computational resources, with a lot of hours spent on Delftblue, to train and evaluate our models. This high computational demand has an environmental impact, primarily due to the energy consumption associated with running large-scale simulations and training deep-learning models. As researchers, we acknowledge this impact and will continue exploring ways to optimize our methods for greater computational efficiency.

To promote transparency and reproducibility, we plan to release our models and training code as open-source resources. This approach ensures that others in the research community can build upon our work, fostering collaborative improvements and enabling educational value. We believe that sharing our models and code openly also supports fairness, as it allows others to validate our results and adapt our techniques for similar challenges.

A potential ethical concern in using RL for control in wind farms is the risk of creating a model that behaves as a "black box," where decisions are challenging to interpret and validate. To mitigate this, we designed our system to incorporate interpretable inputs, such as wind speed maps, which provide insight into how well the model captures complex wake dynamics. By prioritizing interpretability, we aim to develop a control approach that stakeholders can evaluate and trust.

Finally, there is a broader ethical consideration regarding the potential long-term consequences of optimizing for immediate energy output. For example, if our RL controller prioritizes energy production at the expense of structural load on turbines, it may inadvertently lead to higher maintenance costs and reduced turbine lifespans. Furthermore, control decisions could impact local ecosystems or contribute to noise pollution if not properly designed with these external factors in mind. Future work should explore integrating environmental and sustainability factors into the optimization process to ensure our technology aligns with the long-term goals of sustainable and ecologically responsible wind energy.

References

[1] Ronen Basri, David Jacobs, Yoni Kasten, and Shira Kritchman. The convergence rate of neural networks for learned functions of different frequencies, 6 2019.

[2] Yuan Cao, Zhiying Fang, Yue Wu, Ding-Xuan Zhou, and Quanquan Gu. Towards understanding the spectral bias of deep learning, 12 2019.

[3] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. *arXiv (Cornell University)*, 1 2018.

[4] P. A. Fleming, P. M. O. Gebraad, J. W. van Wingerden, S. Lee, M. Churchfield, A. Scholbrock, J. Michalakes, K. Johnson, and P. Moriarty. The sowfa super-controller: A high-fidelity tool for evaluating wind plant control approaches. In *Proceedings of the EWEA Annual Meeting*, 2013.

[5] Scott Fujimoto, Herke Van Hoof, and David Meger. Addressing function approximation error in Actor-Critic methods. *arXiv (Cornell University)*, 1 2018.

[6] Rui Gao and Rajeev K. Jaiman. H-SIREN: Improving implicit neural representations with hyperbolic periodic functions, 10 2024.

[7] P. M. O. Gebraad, F. W. Teeuwisse, J. W. van Wingerden, P. A. Fleming, S. D. Ruben, J. R. Marden, and L. Y. Pao. Wind plant power optimization through yaw control using a parametric model for wake effects – a cfd simulation study. *Wind Energy*, 2014.

[8] Deborah A. Kaminski and Michael K. Jensen. *Introduction to thermal and fluids engineering*. Wiley, 11 2004.

[9] Yongqi Liu, Hui Qin, Zhendong Zhang, Shaoqian Pei, Zhiqiang Jiang, Zhongkai Feng, and Jianzhong Zhou. Probabilistic spatiotemporal wind speed forecasting based on a variational Bayesian deep learning model. *Applied Energy*, 260:114259, 12 2019.

[10] B Moseley. *Physics-informed machine learning: from concepts to real-world applications*. PhD thesis, University of Oxford, 2022.

[11] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron Courville. On the Spectral Bias of Neural Networks, 6 2018.

[12] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 11 2018.

[13] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit Neural Representations with Periodic Activation Functions, 6 2020.

[14] Henk Kaarle Versteeg and Weeratunge Malalasekera. *An introduction to computational fluid dynamics*. Pearson Education, 1 2007.

[15] Huai-Zhi Wang, Gang-Qiang Li, Gui-Bin Wang, Jian-Chun Peng, Hui Jiang, and Yi-Tao Liu. Deep learning based ensemble approach for probabilistic wind power forecasting. *Applied Energy*, 188:56–70, 12 2016.

[16] F. M. White. *Fluid Mechanics*. McGraw-Hill, 5 edition, 2003.

[17] Shengfeng Xu, Chang Yan, Zhenxu Sun, Renfang Huang, Dilong Guo, and Guowei Yang. On the Preprocessing of Physics-informed Neural Networks: How to Better Utilize Data in Fluid Mechanics, 3 2024.

[18] Zhi-Qin John Xu, Yaoyu Zhang, Tao Luo, Yanyang Xiao, and Zheng Ma. Frequency principle: fourier analysis sheds light on deep neural networks. *Communications in Computational Physics*, 28(5):1746–1767, 6 2020.

[19] Jincheng Zhang and Xiaowei Zhao. A novel dynamic wind farm wake model based on deep learning. *Applied Energy*, 277:115552, 7 2020.

[20] Jincheng Zhang and Xiaowei Zhao. Spatiotemporal wind field prediction based on physics-informed deep learning and LIDAR measurements. *Applied Energy*, 288:116641, 2 2021.